ITCS 209 – Object Oriented Programming
Project 02: Orders

```
Update Logs:
2022-03-18      Initial Version (v1.0)
```

Due: Monday, April 25, 2022 11:55PM

Learning Objectives:

On the course of implementing this programming project, you will learn some of the basic concepts of object oriented programming and how to apply them to practical, real world programming applications. Specifically, upon accomplishing this project, we expect you to be able to:

1. Instantiate objects and use them.
2. Apply OOP properties such as encapsulation, inheritance, polymorphism, and interface.
3. Learn how and when to use List and Map data structures.
4. Write a program to read and write text files in simple format as well as in JSON format.
5. Learn how to use regular expressions and simple string manipulation techniques.
6. Get a glimpse of how OOP can be applied to real-world problems.
7. Enjoy coding with Java.

Introduction

You will create a program to handle an order and payment of a store. The store will have several types of items depending on how you design them. A customer can place an order either in-store or online, but they will make an order by sending you an order file. The customer can choose to pay right away or pay later after he/she receive the items. Your program should be able to process these order files and write the order detail for the customer.

Input Order File

The order file is in the format of

Line 1:	Customer Name [(email, zipcode)]
Line 2:	barcode1 quantity1
Line 3:	barcode2 qunatity2
Line …	. . .
Line n:	NONE | Payment Information

The first line is the customer name. For any order from an online customer, there are email and zip code in the parenthesis ( ) as well. The following lines before the last line are the list of items line in the format of barcode and quantity. If the quantity is negative, it means that the customer wants to reduce the items. Finally, the last line contain payment information or none if no such payment yet. For example:

Line 1:	Hesitant Customer
Line 2:	1 20
Line 3:	3 10
Line 4:	2 5
Line 5:	1 -10
Line 6:	1 -5
Line 7:	2 5
Line 8:	1 -10
Line 9:	1 -100
Line 10:	2 -10
Line 11:	NONE

This means only items with the barcode of 3 are left (quantity = 10) without any payment yet.

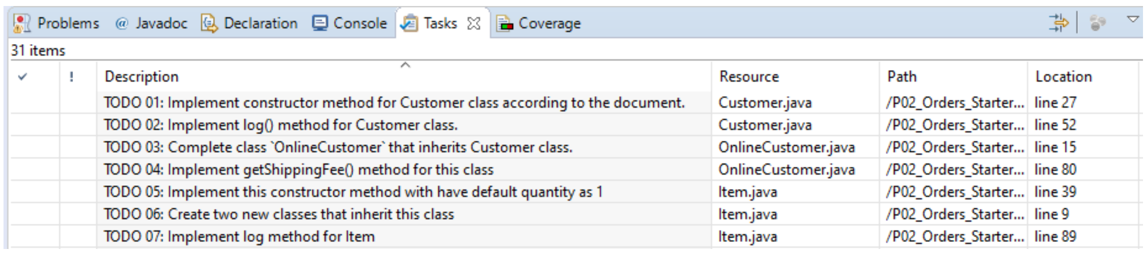
Output Order Summary

The order summary consists of your shop information, customer information, order items, and the total price. An example of the order summary is

Line 1:	Sale Person: Siripen Pongpaichet (6488999)
Line 2:	Customer: 1: Suppawong Tuarob
Line 3:	- Cat Food\t3000.00\t2 (packs)\t6000.00
Line 4:	- Cat Ball Toy\t300.00\t1 (pieces)\t300.00
Line 5:	Total: 6300.00

Instructions

- Read this project **Description** and the **Grading Criteria** carefully.
- Download the zip file "P02_Orders_Starter" from MyCourses.



31 items

✓	!	Description	Resource	Path	Location
		TODO 01: Implement constructor method for Customer class according to the document.	Customer.java	/P02_Orders_Starter...	line 27
		TODO 02: Implement log() method for Customer class.	Customer.java	/P02_Orders_Starter...	line 52
		TODO 03: Complete class `OnlineCustomer` that inherits Customer class.	OnlineCustomer.java	/P02_Orders_Starter...	line 15
		TODO 04: Implement getShippingFee() method for this class	OnlineCustomer.java	/P02_Orders_Starter...	line 80
		TODO 05: Implement this constructor method with have default quantity as 1	Item.java	/P02_Orders_Starter...	line 39
		TODO 06: Create two new classes that inherit this class	Item.java	/P02_Orders_Starter...	line 9
		TODO 07: Implement log method for Item	Item.java	/P02_Orders_Starter...	line 89

- You can start by following the "TODO 1" to "TODO 28". The number here is meant to guide you through this project. In fact, you do not have to strictly follow them.
- Test your code often. There are unit tests for every file with. You will need Junit 5 to runt them. Also, you are encouraged to create your own test cases as you see fit to test your own program. Good programmers run the test to create a bug-free program. Be good! **Note that passing all the test cases does not guarantee full scores. Your program will be tested with another set of test cases for grading purpose.**

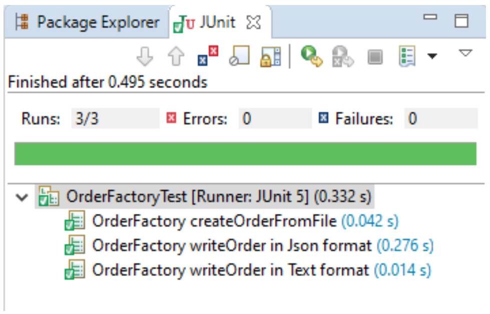
Submission Instructions:

It is important that you follow the submission instructions. **DO NOT** SUBMIT THE WHOLE PROJECT with your system configuration. Failing to do so may result in a deduction and/or delay of your scores. You have to submit 2 things: source code and test result report

1. **Source code submission (xxx.zip file)**
   a. Create a folder and name it P02_<your 7 digit Student ID>. Let's call it the submission folder.
   b. Put the following 10 files into the submission folder (*note that the filenames must be exactly the same as shown here. Otherwise, the autograder will not recognize the files.):
      - Cash.java
      - CreditCard.java
      - Customer.java
      - Item.java
      - ItemFactory.java
      - OnlineCustomer.java
      - Order.java
      - OrderFatory.java
      - Payment.java
      - PaymentFactory.java

   c. [optional] Add any "additional" xxx.java and external .jar library that you used
   ****except gson-2.x.y.jar library – which is already provided for you*****
   d. Zip the folder (e.g., P02_6488999.zip). Make sure that your ID in the file name is correct
   e. [extra step] Redownload the submission and recheck the ID in the submission package, and make sure it is what you wanted to submit. It is your responsibility to submit the correct file.

2. **Test result report (xxx.pdf file)**
   a. Capture the screen of the test report (all the cases)



```
Finished after 0.495 seconds
Runs: 3/3    Errors: 0    Failures: 0
OrderFactoryTest [Runner: JUnit 5] (0.332 s)
    OrderFactory createOrderFromFile (0.042 s)
    OrderFactory writeOrder in Json format (0.276 s)
    OrderFactory writeOrder in Text format (0.014 s)
```

   b. Put all screenshots together in to one single PDF file
   c. Name your file as "P02_<7 digits student ID>_TestReport.pdf

Note: Late submission will suffer a penalty of 20% deduction of the actual scores for each late day. You can keep resubmitting your solutions, but the only latest version will be graded.

Coding Style Guideline:

It is important that you strictly follow this coding style guideline to help us with the grading and for your own benefit when working in groups later in future courses. Failing to follow these guidelines may result in a deduction of your project scores.

1. Write your name, student ID, and section (in commented lines) on top of every Java code file that you submit.
2. Comment your code, especially where you believe other people will have trouble understanding, such as each variable's purposes, loop, and a chunk of code. If you implement a new method, make sure to put an overview comment at the beginning of each method that explains (1) Objective, (2) Inputs - if any, and (3) Output - if any.

Grading Criteria

Some tasks depend on the previous tasks. So, so make sure that you read the TODO description carefully, and work on each task in an appropriate manner.

1.	Follow the submission and coding style guideline	5%
2.	On-time Submission	5%
3.	TODO 01 – 11 about customer and item	20%
4.	TODO 12 – 16 about payment	20%
5.	TODO 18 – 25 about order	30%
6.	TODO 26 – 28 about read/write file	20%
	Total Score	**100%**

[Optional] Bonus for creativity (Max 10%)
You are encouraged to implement your own cool feature for this project (~be creative~) such as creating an interface to accept the input from a user and create the order, making new types of payment method, or colleting multiple orders and doing some data analyzing such as finding top-10 best seller products or finding which items are usually bought together using *Market Basket Analysis*[1].

There is a separate submission for the bonus submission on MyCourses. In this submission, you have to submit your new source code and at most two pages report explaining what you do and how to run/test your implementation.

{ Good luck, and enjoy coding }

[1] https://www.kdnuggets.com/2019/12/market-basket-analysis.html

Bug Report:
Though not likely, it is possible that our solutions may contain bugs. In this context, a bug is not an insect, but an error in the solution code that we implemented to generate the test cases. Hence, if you believe that your implementation is correct, yet yields different results, please contact us immediately so proper actions can be taken.

Need help with the project?

If you have questions about the project, please first <u>ask them on the class' general channel on Teams</u>, so that other students with similar questions can benefit from the discussions. The TAs can also answer some of the questions and help to debug trivial errors. If you still have questions or concerns, please make an appointment with one of the instructors. ***We do not debug your code via email***. If you need help with debugging your code, please come see us.

Academic Integrity (Very Important)

Do not get bored about these warnings yet. But please, please ***do your own work***. Your survival in the subsequent courses and the ability to get desirable jobs (once you graduate) heavily depend on the skills that you harvest in this course. Though students are allowed and encouraged to discuss ideas with others, the actual solutions must be originated and written by themselves. Collaboration in writing solutions is not allowed, as it would be unfair to other students. Students who know how to obtain the solutions are encouraged to help others by guiding them and teaching them the core material needed to complete the project, rather than giving away the solutions. ***You can't keep helping your friends forever, so you would do them a favor by allowing them to be better problem solvers and life-long learners.*** Your code will be compared with other students' (both current and previous course takers) and online sources using state-of-the-art source-code similarity detection algorithms which have been proven to be quite accurate. If you are caught cheating, serious actions will be taken, and heavy penalties will be bestowed on all involved parties, including inappropriate help givers, receivers, and middlemen.